TODO: incluir portada oficial

Categories in Homotopy Type Theory

(Título provisional - busca uno definitivo - probablemente en mayo)

Pedro Bonilla Nadal

Pedro Bonilla Nadal *Categories in Homotopy Type Theory*

.

Master's thesis.
Academic year 2020-2021.

Thesis supervisor Dr. Ángel González Prieto
Departamento - Universidad

Máster en Matemáticas y Aplicaciones

Universidad Autónoma de Madrid

Contents

Part I

Category Theory

Chapter 1

First Notions of Category Theory

> "The human mind has never invented a labor-saving machine equal to algebra."
>
> *Stephan Banach (1925)*

For us categories are an area of interest on its own rights, rather than merely an elegant tool. In this sense we will introduce this theory with a point of view, trying to emphasize the intuitive ideas behind each notion.

Thus, in addition to the formal content, we will also provide enough information so that an avid reader gain quick intuitive understanding of the subject. With this objective in mind, we will get into the habit of introducing the first examples even before introducing the formal definitions. Thus, when the formal definition is introduced, it becomes more meaningful and helps the intuition to understand the concept.

The fundamental idea of Category Theory is that many properties can be unified if expressed with diagram and arrows. Intuitively, a diagram is a directed graph, such that each way of going from a node to another are equals. For example, the diagram:

$$\begin{tikzcd} & b \arrow[dr, "g"] & \\ a \arrow[ur, "f"] \arrow[rr, dashed, "h"] & & c \end{tikzcd}$$

Means that $f \circ g = h$. This approach to mathematics emphasises at the relationships between elements, rather than at the elements themselves (and from them derive their relationships). In general, dashed lines means that the existence of that particular arrow is uniquely determined by the solid arrow presents in the diagram.

In this first chapter we will introduce the notion of category and their first properties. The principal references for this chapter are [3] and [5]. Haskell references can be checked in [4].

1.1 Metacategories

We will start by defining a concept independent of the set theory axioms: the concept of *metacategory*. Once this is done we will follow reinterpreting this definitions in the context of set theory. We follow [3] for these definitions.

Traditionally, mathematics is based on the set theory. When we start set theory it is not necessary (it is not possible) to define what a set is. It is similar with the concepts of element and belonging, which are basic to set theory. Category theory can also be used to found mathematics. In this sense we will give definitions based on other concepts such as object, arrow, composition.

Definition 1.1.1. A *metagraph* consist of *objects*: $a, b, c..$ and *arrows* $f, g, h....$ There are also two pairings: dom and codom. This pairings assigns each arrow with an object. An arrow f with $\mathsf{dom}(f) = a$ and $\mathsf{codom}(f) = b$ is usually denoted as $f : a \to b$.

Definition 1.1.2. A metacategory is a metagraph with two additional operations:

- *Identity*: assigns to each object a an arrow $1_a : a \to a$.
- *Composition*: assigns to each pair of arrows f, g with $\mathsf{codom}(f) = \mathsf{dom}(g)$ and arrow $g \circ f$ such that the diagram:

$$\begin{tikzcd} & b \arrow[dr, "g"] & \\ a \arrow[ur, "f"] \arrow[rr, dashed, "g\circ f"'] & & c \end{tikzcd}$$

  commutes. The arrow $g \circ f$ is called the *composite* of f and g.

There are additional two properties:

- *Associative*: given arrows f, g, h, we have that,

$$(g \circ f) \circ h = g \circ (f \circ h).$$

- *Unit*: given an object a, and arrows f, g such that $\mathsf{dom}(f) = a$ and $(g) = a$, we have that,

$$1_a \circ g = g, \qquad f \circ 1_a = f.$$

In the context of a metacategory, arrows are often called *morphisms*.

We have just define what a metacategory is without any need of set and elements. In most cases we will rely in a set theory interpretation of this definitions, as most examples will rely on this theory. Nonetheless, whenever possible, we will define the concepts working only in terms of objects and arrows, having therefore the theory as independent as possible to this theory.

1.2 ZFC Axioms

(Maybe) TODO: Introduce fundamental terminology and problems related to classical set theory.

Once we have talk so much about this we may also formally introduce the stuff. Is not so long (?) - ok, maybe it is. I am letting this as a TODO and continue with other stuff until this is more done.

Further on i left a detailed TODO of things from this section that are referenced in the text.

Detailed TODO:

- small sets and large sets.

1.3 Set theory categories

Further on we will work based on set theory. Despite that, it is useful to take into account that most of the concept here presented can be defined without making use of this, To act accordingly if needed. However, for our interests we will never need to get rid of set theory.

Definition 1.3.1. A category (resp. graph) is an interpretation of a metacategory (resp. metagraph) within set theory.

That is, a graph/category is a pair (O, A) where a collection O consisting of all objects as well as a collection A consisting of all arrows. Their elements holds the same properties that objects and arrows satisfy on metacategories / metagraph.

We will focus on the category. We can also define the function homeset of a category $C = (O, A)$, wrote as $\hom_C$, as the function:

$$\begin{aligned} \hom_C : O \times O &\mapsto \mathcal{P}(A) \\ (a, b) &\mapsto \{f \in A | f : a \to b\} \end{aligned}$$

When there is no possibility of confusion we will state $c \in C$ without specifying . We will refer to the collection of objects of a category C as $Ob(C)$ and the collection of arrows as $Ar(C)$.

Definition 1.3.2. We say that a category is *small* if the collection of objects is given by a set (instead of a proper class). We say that a category is *locally small* if every homesets is a set.

We proceed to introduce a comprehensive list of examples, so that it is already introduced in subsequent chapters.

Example 1.3.1.

- The elementary categories:
  - The category $\mathbf{0} = (\emptyset, \emptyset)$ where every property is trivially satisfy.
  - The category $\mathbf{1} = (\{e\}, \{1_e\})$.
  - The category $\mathbf{2} = (\{a, b\}, \{1_a, 1_b, f : a \to b\})$
- Discrete categories: are categories where every arrow is an identity arrow. This are sets regarded as categories, in the following sense: every discrete category $C = (A, \{1_a : a \in A\})$ is fully identified by its set of object.
- Monoids and Groups: A monoid is a category with one object (regarding the monoid of the arrows). In the same way, if we requires the arrows to satisfy the inverse property, we can see a group as a single-object category.
- Preorder: From a preorder $(A, \leq)$ we can define a category $C = (A, B)$ where B has an arrow $e : a \to b$ for every $a, b \in A$ such that $a \leq B$. The identity arrow is the arrow that arise from the reflexive property of the preorders.
- Large categories: these categories has a large set of objects. For example:
  - The category Top that has as objects all small topological spaces and as arrow continuous mappings.
  - The category Set that has as objects all small sets and as arrows all functions between them. We can also consider the category Set_* of pointed small sets (sets with a distinguish point), and function between them that maps one distinguish point into another.
  - The category $Vect$ That has as object all small vector spaces and as arrows all linear functions.
  - The category Grp That has as object all small vector group and as arrows all homomorphism.
  - The category Top_* that has as object all small topological spaces with a distinguished point, and as arrow all continuous functions that maps each distinguished points into distinguished points. Similarly we can consider Set_* or Grp_*.
- The category $Hask$ of all Haskell types and all possible functions between two types.

Note that, for example, as natural numbers can be seen as either a set or a preorder, they also can be seen as a discrete category or a preorder category.

Simple enough categories can be easily described with diagrams: there are as many objects as nodes in the diagram, and an arrow in the category for each arrow in the diagram, composition of existing arrows, and every identity arrow (that are usually omitted).

For example, we can fully represent the $\mathbf{2}$ category with the diagram:

$$a \xrightarrow{g} c$$

1.3.1 Properties

We can see that is common in mathematics to have an object of study (propositional logic clauses, groups, Banach spaces or types in Haskell). Once the purpose of studying these particular sets of objects is fixed, it is also common to proceed to consider the transformations between these objects (partial truth assignments, homomorphisms, linear bounded functionals or functions in Haskell).

In categories, we have a kind of different approach to the subject. Instead of focusing on the objects themselves, we focus on how they relate to each other. That is, we focus on the study of the arrows and how they composes. Therefore we can consider equal two objects that has the same relations with other objects. This inspire the next definition:

Definition 1.3.3. [5, Definition 1.1.9] Given a category $C = (O, A)$, a morphism $f : a \to b \in A$ has a *left inverse* (resp. *right inverse*) if there exists a $g : b \to a \in A$ such that $g \circ f = 1_b$ (resp. $f \circ g = 1_a$). A morpishm is an *isomorphism* if it has both left and right inverse, called the *inverse*. Two object are isomorphic is there exists an isomorphism between them.

Is easy to follow that this functions define bijections between $\hom(a, c)$ and $hom(b, c)$ for all $c \in O$. Also one can see that if a morphism has a left and a right inverse, they must be the same, thus implying the uniqueness of the inverse.

We will now proceed to talk about certain arrows and objects that have properties that distinguish them from others. To accompany and gain an intuition of these properties, I find the most useful example is that of the $FinSet$ category, of all finite sets and functions between them. Considering special arrows:

Definition 1.3.4. An arrow f is *monic* (resp. *epic*) if it is left-cancelable (resp. right-cancelable), i.e. $f \circ g = f \circ h \implies g = h$ (resp. $g \circ f = h \circ f \implies g = h$).

In FinSet this arrows are the injective functions (resp. surjective). Considering special objects:

Definition 1.3.5. an object a is *terminal* (resp. *initial*) if for every object b there exists an unique arrow $f : b \to a$ (resp. $f : a \to b$). An object that is both terminal and initial is called *zero*.

In FinSet this objects is the initial object is the empty set and the terminal object the one point set. In the Pointed

Proposition 1.3.1. Every two terminal object are isomorphic.

Proof. Every terminal object has only one arrow from itself to itself, and necessarily this arrow has to be the identity. Let a, b be terminal object and $f : a \to b$ and $g : b \to a$ be the only arrows with that domain and codomain. Then $f \circ g : a \to a \implies f \circ g = 1_a$. Analogously $g \circ f = 1_b$.

□

Another important property in category theory is the *duality property*. In sort, this property tell us that for every theorem that we prove for categories, there exist another theorem that is automatically also true. To prove this, we define the concept of *opposite category*.

Definition 1.3.6. [5, Definition 1.2.1] Let C be any category. The opposite category C^{op} has:

- the same objects as in C,
- an arrow $f^{op} \in C^{op}$ for each arrow $f \in C$, so that the domain of f^{op} is defined as the codomain of f and viceversa.

The remaining structure of the category C^{op} is given as follows:

- For each object a, the arrow 1_a^{op} serves as its identity in C^{op}.
- We observe that f^{op} and g^{op} are composable when f and g are, and define $f^{op} \circ g^{op} = (g \circ f)^{op}$

The intuition is that we have the same category, only that all arrows are turned around. We can see that for each theorem T that we prove, we have reinterpretation that theorem to the opposite category. Intuitively this theorem is an equal theorem in which all the arrows have been turned around. For example: the proposition 1.3.1 can be reworked as:

Proposition 1.3.2. Every two initial object are isomorphic.

That is because being initial is the dual property of being terminal, that is, if $f \in C$ is a terminal object then $f^{op} \in C^{op}$ is an initial object. Being isomorphic is its own dual.

1.3.2 Transformation in categories

This is one of the main ways of defining a category: consider a collection of objects and the standard way of transforms one into each other. Then, we may also follow our study defining the structure preserving transformation of categories.

Definition 1.3.7. Given two categories C, B, a *functor* $F : B \to C$ is a pair of functions $F = (F' : Ob(C) \to Ob(B), F'' : Ar(C) \to Ar(B))$ (the *object function* and the *arrow functor* respectively) in such a way that:

$$F''(1c) = 1_{F'c},\ \forall c \in Ob(C); \qquad F''(f \circ g) = F''f \circ F''g,\ \forall f, g \in Ar(C).$$

That is, a functor is a morphism of categories. When there is no ambiguity we will represent both F' and F'' with a single symbol F acting on both objects and arrows. Also, as you can see in the definition, whenever possible the parentheses of the functor will be dropped. This loss of parentheses will be replicated thorough the text, whenever possible.

An important consideration on functors is that a functor F can be defined only pointing out how it maps arrows (assuming that such mapping respects the composition), as how F maps object can be defined with how it map identity arrows.

Lets provide some examples of functors:

Example 1.3.2.

- Forgetfull functor: We have a variety of categories consisting of structures on sets such as objects and functions that hold structure such as arrows (eg Top, Grp or Vect).

  Let C one of such categories, then we have a functor $F : C \to Set$ that maps each object to its underlying set and each arrow to the equivalent arrow between sets. That is, this functor forgets about the structure that is present in C.

  Conversely, we will often have functors defined $F : C \to Set$ for C any category. On some cases of C it also happen that FC has some more structure on it. In that case we will say that F is an enriched functor for that cases.

- Fundamental Group: In the context of algebraic topology we have the functor of the fundamental group $\Pi_1 : Top_* \to Grp$. The most famous property of this function is that

  > Given continuous application $f, g : X \to Y$ between two topological spaces induces an application of the group of loops of X on the group of loops of Y such that $\Pi_1(f \circ g) = \Pi_1(f) \circ \Pi_1(g)$.

  that is, the functoriality of the function (taking also into account the fact that the identity of topological spaces is mapped to the identity of group).

- Stone-Čech Compactification[1]: In the realm of functional análisis we have the following theorem:

[1] https://en.wikipedia.org/wiki7Stone\T1\textendashČech_compactification#Universal_property_and_functoriality

Theorem 1.3.1. [2] *Let Ω be a completely regular Hausdorff topological space. Then there exists a compact Hausdorff topological space Ω and a homeomorphism Φ, from Ω to a dense subset of Ω. Moreover, for every continuous and bounded function $f : \Omega \to \mathbb{K}$ there exists a unique $\overline{f} \in C(\beta\Omega)$ such that $\overline{f} \circ \Phi = f$ and the application $f \to \overline{f}$ is an isometric isomorphism. Finally, $\beta\Omega$ is uniquely determined up to isomorphisms by the above properties.*

Proof. It is enough to take as the closure of $\Delta(\Omega)$ in the induced weak-topology ω^* (since in the weak topology we have that a bounded closed is compact).

Given a continuous and bounded function $f : \Omega \to K$ we can define the function $\overline{f}$ by injection of Ω and then limit to maintain continuity (which will exist by compactness and boundedness of f). Thus the maximum of $overlinef$ can be reached by a sequence of points injected from $Omega$ obtaining that

$$\max \left\{ \mid \overline{f}\ (s) \mid : s \in \beta\Omega \right\} = \sup \{|f(s)| : s \in \Omega\}.$$

The application $f \to \overline{f}$ is therefore an isometric isomorphism. Thanks to this isometric isomorphism the function spaces and by the Banach-Stone theorem[1, Theorem 3] we have uniqueness up to isomorphisms. □

Question: I want to include this example of functor in order to later reuse it in universality section (opening of chapter 2).

I'm having a bit of trouble doing this though, as I do not see how does it maps arrows. I have found this proof of its functoriality:

> As is usual for universal constructions like this, the extension property makes a functor from *Top* (the category of topological spaces) to *CHaus* (the category of compact Hausdorff spaces).

However, I was hoping to provide a more elemental proof, so I will continue looking. If nothing is founded, there will be no problem to drop this example, or to explain directly when extension property is explained. End Question

- Group actions: Every group action can be seen as a functor from a Group to a Set.

Let G be a group seen as a category with only one element and S be a set seen as a discrete category. Then, an action is a representation of the group of the endomorphism of the set, that is, an action α associate each

[2] I have a little prove of this theorem that uses the Banach-Stone theorem, but was a bit hesitant about whther should I include it.

element of the groupwith an function $\alpha(g) : X \to X$ such that the product of element of the group is maintain via composition. That is for $g \in Ar(G)$ we have that $\alpha(g) \in Ar(X)$ and for

$$\alpha(g \circ^1 f) = \alpha(g) \circ \alpha(f) \qquad \forall g, f \in Ar(G),$$

where in 1. it is to note that is a group seen as a category the product is the composition) thus having the functoriality.

- `Maybe` Method in Haskell [4, Section 7.1]): In Haskell the definition of Maybe is a mapping from type a to type $\mathrm{Maybe}\, a$.

```
data Maybe a = Nothing | Just a
```

LISTING 1.1: Declaration of Maybe

Note that `Maybe a` is not a type but a function of types. In order for it to be an endofunctor of the *Hask* category we will need for it to also map function. Let $f : a \to b \in Hask$, then we define the function

$$\mathrm{Maybe}\, f : \mathrm{Maybe}\, a \to \mathrm{Maybe}\, b$$

such that

$$(\mathrm{Maybe}\, f)(Nothing) = Nothing, \qquad (\mathrm{Maybe}\, f)(Just\ a) = Just\ f(a).$$

Note that $(\mathrm{Maybe}\, id_a)(\mathrm{Maybe}\, a) = Nothing \mid Just\ id_a(a) = \mathrm{Maybe}\, a$.

We can now construct the category of all small categories Cat. This category has as object all small categories and as arrows all functor between them. Note that Cat does not contain itself.

We can consider some properties in functors. Before defining then, note that we can consider a functor $T : C \to D$ as a function over homeset, that is, a function:

$$T : \mathrm{hom}_C(a, b) \to \{Tf : Ta \to Tb | f \in C\} \subset \mathrm{hom}_D(Ta, Tb)$$

Definition 1.3.8. A functor $T : C \to D$ is *full* if it is surjective as a function over homesets, i.e. if the function $T : \mathrm{hom}_C(a, b) \to \mathrm{hom}_D(Ta, Tb)$ is surjective for every $a, b \in C$. A functor is *faithfull* if it is injective over homesets.

As we have defined the concept of opposite category, we can consider functors $T : C^{op} \to D$. This functor, is called a *contravariant* functor from C to D, in oposition of a covariant functor from C to D. A functor $T : C \to D^{op}$ is usually called a *covariant* functor from C to D.

We shall continue defining natural transformation. In the words of Saunders Mac Lane:

> Category has been defined in order to define functor and functor has been defined in order to define natural transformation.

One can see a functor $T : C \to D$ as representation of a category in another, in the sense that a functor provide a picture of the category C in D. Further elaborating into this idea, we can consider how to transform these drawings into each other.

Definition 1.3.9. Given two functor $T, S : C \to D$, a *natural transformation* $\tau : T \Rightarrow S$ is a function from $Ob(C)$ to $Ar(D)$ such that for every arrow $f : c \to c' \in C$ the following diagram:

$$\begin{tikzcd} Tc \arrow[r, "\tau c"] \arrow[d, "Tf"] & Sc \arrow[d, "Sf"] \\ Tc' \arrow[r, "\tau c'"] & Sc' \end{tikzcd}$$

commutes. A natural transformation where every τc is invertible is called a *natural equivalence* and the functors are *naturally isomorphic*.

That is a natural transformation is a map between pictures of C into D. Note that a natural transformation acts only on the domain of objects. Lets provide some examples.

With the same notation as in the definition, when we have to define natural transformations it will often be useful to define each arrow individually. In this case we will note $\tau(c) = \tau_c : Tc \to Sc$.

Example 1.3.3.

- The opposite group: We can define the opposite functor $(\cdot)^{op} : Grp \to Grp$ that maps $(G, *)$ to $(G; *^{op})$ where $a *^{op} b = b * a$ and map a morphism $f : G \to G'$ we define $f^{op}(a) = f(a)$ and have that

$$f^{op}(a *^{op} b) = f(b * a) = f(b) * f(a) = f^{op}(a) *^{op} f^{op}(b).$$

  Denoting the identity function $Id : Grp \to Grp$, we have a natural transformation $\tau : Id \Rightarrow (\cdot)^{op}$ defined by $\tau_G(a) = a^{-1}$ for all $a \in G$.

- In Haskell, the safeHead [4, Section 10.1] function between the List functor and the Maybe functor. Lets start by defining the functors:
  - We can define the list functor that maps the type a to the type $[a]$ and apply each function $f : a \to b$ to $f' : [a] \to [b]$ that applies f elements wise (on empty list it does nothing).

```
data List a = Nil | Cons a ( List a)
```

  - We can define the natural transformation SafeHead : List $\to$ Maybe defined as follow:

```
safeHead :: [a] -> Maybe a
safeHead [] = Nothing
safeHead (x : xs) = Just x
```

1.3.3 Constructions

In this last subsection we will introduce some standard construction on categories, along with some examples of these constructions.

Product Category

We present now one of the most usual construction in mathematics: the product. We will consider the "universal" properties of product on the next chapter. By now, we present the product of categories.

Definition 1.3.10. Let B, C be categories. Then the *product category* $B \times C$ is the category that has as objects the pairs $\{< b, c >: b \in Ob(B), c \in Ob(C)\}$ and as arrows the pairs of arrows $\{< f, g >: f \in Ar(B), g \in Ar(C)\}$. The composition of arrows if defined by the elementwise composition.

It is clear that we can define to functor $P : B \times C \to B$ and $Q : B \times C \to C$ that restricts the category to each of its component parts (functorial axioms follow immediately). Moreover, we can see that any functor $F : D \to B \times C$ will be uniquely identified by its composition by P and Q.

Complementary, for any two functors $F : D \to B, G : D \to C$ we can define an functor $F \times G : D \to B \times C$ that apply $(F \times G) < f, g >=< Fg, Gg >$. Expressed as a diagram:

$$\begin{tikzcd}
& D \arrow[dl, "F"'] \arrow[d, dashed, "F\times G"] \arrow[dr, "G"] & \\
B & B \times C \arrow[l, "P"] \arrow[r, "Q"'] & C
\end{tikzcd}$$

A functor $F : B^{op} \times C \to D$ is called *bifunctors*. Arguably the most important bifunctor is the $\hom_C$ function seen as a functor. Given a category C we can see $\hom_C : C^{op} \times C \to Set$ as a bifunctor such that:

$$\hom_C(\cdot, \cdot)(a, b) = \hom_C(a, b) \qquad \forall a, b \in Ob(C)$$

for the object. For the arrows:

$$\begin{aligned} \hom_C(f^{op}, g) : \hom_C(a', b) \to \hom_C(a, b') &\qquad \forall f : a \to a', g : b \to b' \in C \\ \hom_C(f^{op}, g)(h) = g \circ h \circ f &\qquad \forall h \in \hom_C(a', b) \end{aligned}$$

From this bifunctor we can define two functors for every $c \in Ob(C)$:

- The functor $\hom_C(c, \cdot) : C \to Set$ such that

$$\hom_C(c, \cdot) = \hom_C(c, d) \qquad \forall d \in Ob(C)$$

  for the object. For the arrows:

$$\begin{aligned} \hom_C(c, g) : \hom_C(a, d) \to \hom_C(a, d') &\qquad \forall g : d \to d' \in C \\ \hom_C(c, g)(h) = g \circ h &\qquad \forall h \in \hom_C(c, d) \end{aligned}$$

- The covariant functor $\hom_C(\cdot, c) : C^{op} \to Set$, defined analogously.

Functor Categories

We continue defining functor categories, that is, categories where we consider the functors as objects and natural transformation as arrows in some sense. This concept will be instrumental in further consideration in the realm of functional programming (in particular, in the definition of monad).

Before explaining composition between natural transformation, it is useful to present the following diagram. Let B, C be categories, $F, G : B \to C$ be functors and $\tau : F \to G$ natural transformation τ, it is common to represent this structure with:

$$\begin{tikzcd} A \arrow[r, bend left=50, "F"] \arrow[r, bend right=50, "G"'] \arrow[r, phantom, "\Downarrow \tau"] & B \end{tikzcd}$$

Lets define the composition of two natural transformation. We will start defining the composition of two natural tranformations σ, τ as in:

$$\begin{tikzcd} A \arrow[r, bend left=60, "F"] \arrow[r, "G"] \arrow[r, bend right=60, "H"'] \arrow[r, phantom, shift left=3, "\tau \Downarrow"] \arrow[r, phantom, shift right=3, "\sigma \Downarrow"] & B \end{tikzcd}$$

Due to this representation, this composition of natural transformations is called *vertical composition*, in opposition to the *horizontal composition* (def. 1.3.13).

Definition 1.3.11. Let C and B be two categories, $R, S, T : C \to B$ be functors, and let $\tau : R \to S$, $\sigma : S \to T$, we define the composition $(\tau \circ \sigma)c = \tau c \circ \sigma c$.

To see that $(\tau \circ \sigma)$ is a natural transformation it suffices the following diagram[6]:

$$
\begin{tikzcd}
Rc \arrow[r, "Rf"] \arrow[d, "\sigma c"'] \arrow[dd, bend right] & Rc' \arrow[d, "\sigma c'"] \arrow[dd, bend left] \\
Sc \arrow[r, "Sf"] \arrow[d, "\tau c"'] & S(c') \arrow[d, "\tau c'"] \\
Tc \arrow[r, "Tf"] & Tc'
\end{tikzcd}
$$

Definition 1.3.12. Let B, C be categories. We define the category B^C as the functor category from C to B, that is, the category with all functors $F : C \to B$ as object, natural transformation as arrows, and composition as defined in 1.3.11.

we now present some examples to provide some intuition about when this type of construction are considered.

Example 1.3.4.

- The category of group action over a set: As we have seen in 1.3.2 each group action over a set is a functor. Let G be a group and S be a set, both seen as categories. Then we can consider the category S^G, that has as object every group action of G over S and as arrow the morpishm of actions.

- C^2: In 1.3.1 we define the 2 category and can consider therefore the category of functor from 2 to C. This category is called the arrow category, as the can see that there are a functor from 2 to C as functor for each arrow in C and conversely.

  This example display a interesting idea: we can consider small collection of object as functions/functors. This idea can as well be seen when we define a pair of real numbers as any function $f : \{0, 1\} \to \mathbb{R}$. Should we want, for example, to consider all the squares in a category C, we might ass well define the category square Sqr and consider every functor $F : Sqr \to C$.

$$
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "g"] & a' \arrow[d, "h"] \\
b \arrow[r, "k"] & b'
\end{tikzcd}
$$

The square category.

  Note that the elements $F : Sqr \to C$ are the arrows of the arrow category of C.

- We can use this construction to study the category C^C of endofunctors of a particular category C. This case is particularly interesting while studying the endofunctors present in $Hask^{Hask}$ and later consider the Monad as a programming structure.

Note that this is not the only way in which to define the composition of natural tranformation. In fact, we can define another functor category. In this case we compose two natural transformation as in:

$$\begin{tikzcd} B \arrow[r, bend left, "F"] \arrow[r, bend right, "G"'] \arrow[r, phantom, "\Downarrow\tau"] & C \arrow[r, bend left, "F'"] \arrow[r, bend right, "G'"'] \arrow[r, phantom, "\Downarrow\sigma"] & D \end{tikzcd}$$

Lets formalize this composition:

Definition 1.3.13. Let B, C, D be categories, $F, G : B \to C, F', G' : C \to D$ be functors, and let $\tau : F \to G$, $\sigma : F' \to G'$, we define the composition $(\tau \circ \sigma) : F \circ F' \to G \circ G'$ such that

$$(\sigma \circ \tau)c = Fc \circ G'\tau c$$

for all $c \in B$.

With this horizontal composition we can properly defined.In this case we can see that the composition of two natural transformation is indeed a natural transformation due to the commutativity of:

$$\begin{tikzcd} F'Fc \arrow[r, "\sigma Fc"] \arrow[d, "F'\tau c"] \arrow[dr, "(\sigma\circ\tau)c"] & G'Fc \arrow[d, "G'\tau c"] \\ F'Gc \arrow[r, "\sigma Gc'"] & G'Gc' \end{tikzcd}$$

With this definition of composition we can consider another different category: the category of all functors of all (small)[3] categories, that is, the category that has all functors as object, and has the natural transformation with horizontal composition as arrows.

When we have to consider both compositions at the same time we denote the vertical composition with $\tau \cdot \sigma$ and horizontal composition with $\tau \circ \sigma$, as in [3]. Lastly we have to consider how this composition relate to each other. This is seen in the *interchange law*:

Proposition 1.3.3. Let A, B, C be categories, $F, G, H : A \to B$, $F', G', H' : B \to C$ be functors and $\tau : F \to G, \sigma : G \to H, \tau' : F' \to G', \sigma' : G' \to H'$ be natural transformations, then:

$$(\sigma' \circ \sigma) \cdot (\tau' \circ \tau) = (\sigma' \cdot \tau') \circ (\sigma \cdot \tau)$$

[3]me parece que tengo que incluir esto pero lo hago más por miedo que por conocimiento

Proof. We have a structure like

$$\begin{tikzcd}
A \arrow[r, bend left=60, "F"] \arrow[r, "G"] \arrow[r, bend right=60, "H"'] \arrow[r, phantom, shift left=3ex, "\tau \Downarrow"] \arrow[r, phantom, shift right=3ex, "\sigma \Downarrow"] & B \arrow[r, bend left=60, "F'"] \arrow[r, "G'"] \arrow[r, bend right=60, "H'"'] \arrow[r, phantom, shift left=3ex, "\tau' \Downarrow"] \arrow[r, phantom, shift right=3ex, "\sigma' \Downarrow"] & C
\end{tikzcd}$$

From the naturality of τ' we have that for all $c \in A$:

$$\begin{aligned}
((\sigma' \cdot \tau') \circ (\sigma \cdot \tau))(c) &= H'((\sigma \cdot \tau)c) \circ (\sigma' \cdot \tau')(Fc) \\
&= H'(\sigma c \circ \tau c) \circ (\sigma'(Fc) \circ \tau'(Fc)) \\
&= H'(\sigma c) \circ H'(\tau c) \circ \sigma'(Fc) \circ \tau'(Fc) \\
&= H'(\sigma c) \circ \sigma' Gc \circ G' \tau c \circ \tau'(Fc) \\
&= (\sigma' \circ \sigma)(c) \circ (\tau \circ \tau)(c) \\
&= ((\sigma' \circ \sigma) \cdot (\tau' \circ \tau))(c).
\end{aligned}$$

□

To finally consider the relation between product, and the functor category we can see that given 3 small categories A, B, C we have a bijection:

$$\hom_{Cat}(A \times B, C) \cong \hom_{Cat}(A, C^B).$$

This fact will be taken into account further in the text, as this will mean that the functor $\cdot \times B : Cat \to Cat$ has a right adjoint.

Comma Category

To define the comma category, we define the Category $(b \downarrow S)$ of object S-under b, sketch the dual notion, and generalize this two concepts with the comma category.

Given a functor $F : B \to C$ an object $b \in B$ is $F-$under another object $c \in C$ if there exists an arrow $f : c \to Fb \in C$. This can be represented as

$$\begin{tikzcd}
c \arrow[d, "f"] \\
Fb
\end{tikzcd}$$

and thus the name of F-under.

Definition 1.3.14. Let B, C be (small)[4] categories and $S : B \to C$ be a functor. For every $c \in Ob(C)$ we can define the category $(c \downarrow S)$ that has as objects all pairs $(b, f) \in Ob(B) \times Ar(C)$ such that $f : c \to Sb$ and as arrows $h : (d, f) \to (d', f')$ every arrow $h : d \to d' \in Ar(B)$ such that $f' = Sh \circ f$.

[4]check this

The property that each arrow should satisfy can be represented as:

$$\begin{tikzcd} & c \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ Sd \arrow[rr, "Sh"'] & & Sd' \end{tikzcd}$$

Example 1.3.5. Let $U : Grp \to Set$ be the forgetful functor, and let $X \in Ob(Set)$. We can consider $(X \downarrow U)$ where every object is a function $f : X \to Ug$ for a group g.

One can easily now deduce the dual concept of the category $(b \uparrow S)$ represented by:

$$\text{objects: } \begin{tikzcd} c \\ Fb \arrow[u, "f"] \end{tikzcd} ; \qquad \text{arrows: } \begin{tikzcd} & c & \\ Sd \arrow[ur, "f"] \arrow[rr, "Sh"'] & & Sd' \arrow[ul, "f'"'] \end{tikzcd} .$$

Now suppose that we have three categories B, C, D and to functors S, T such that

$$B \xrightarrow{S} C \xleftarrow{T} D$$

me might want to consider the relations objects of B and D, for that we have the comma category:

Definition 1.3.15. Let B, C, D be categories and to $S : B \to C, T : C \to D$. We define the coma category $(S \downarrow T)$ as the category that has as object the triples (b, d, f) with $b \in Ob(B), d \in Ob(D), f : Sb \to Td \in Ar(C)$ and arrows the pairs $(g, h) : (b, d, f) \to (b', d', f')$ with $g : b \to b' \in Ar(B), h : d \to d' \in Ar(D)$ such that $Th \circ f = f' \circ Sg$.

We can represent the previous definition by:

$$\text{objects: } \begin{tikzcd} Sb \arrow[d, "f"] \\ Td \end{tikzcd} ; \qquad \text{arrows: } \begin{tikzcd} Sb \arrow[r, "Sg"] \arrow[d, "f"] & Sb' \arrow[d, "f'"] \\ Td \arrow[r, "Sh"] & Td' \end{tikzcd} .$$

The name comma category comes from is alternative notation $(S \downarrow T) = (S, T)$. We prefer the $(S \downarrow T)$ as it is more clear, nonetheless it is clear that before modern text editor became popular, the original comma notation had a big plus.

Chapter 2

Universality, Adjoints and Closed Cartesian Categories

TODO Small paragraph introducing the notions, and sketching a brief summary of the chapter.

2.1 Universality

In this section we present the concept of universality. This concept is behind lots of mathematical properties. Intuitively, universality is an efficient way of expressing an one-to-one correspondence between arrows of different categories. This one-to-one relationship is usually expressed via "given an arrow y it exists one and only one arrow x such that <insert your favorite universal property>".

Prior to the formal definition, we shall introduce an example. Probably the first contact that any mathematician has with universality is when we first try to define a function $f : \mathbb{R} \to \mathbb{R}^2$. We quickly understand that defining such a function is equivalent to define two $g, h : \mathbb{R} \to \mathbb{R}$(we further explain the product in 2.1.6). This uniqueness is the flavor that attempts to capture the concept of universality. Other examples of unique existence are those that occur in quotient groups or in bases of vector spaces. This example will be further formalized after the definition.

Definition 2.1.1. Let $S : D \to C$ be a functor and $c \in Ob(C)$, an *universal arrow* from c to S is a pair (d, u) with $d \in Ob(D), u : c \to Sd \in Ar(C)$, such that for every (e, f) with $e \in Ob(D)$ and $f : c \to Sd$ there exists an unique $f' : r, d \in Ar(D)$ such that $u \circ Sf' = f$.

red Todo, universality from a functor. In a diagram:

$$\begin{tikzcd} c \arrow[r, "u"] \arrow[dr, "f"'] & Sd \arrow[d, dashed, "Sf'"] & d \arrow[d, dashed, "f'"] \\ & Se & e \end{tikzcd}$$

Note that an universal arrow (d, u) induces the unique existence of an arrow in D, but with interesting properties via it relationship with S. Whenever possible to provide an universal arrow we will only define the functor $S : D \to C$ and the arrow $u : c \to Sr$, letting all other information be deduced from the context.

Lets formalize the prior examples and provide some more:

Example 2.1.1.

- Quotient Group:

  From this property alone the three isomorphism theorems can be deduced. Therefore, we only have to prove this result to have the full power of these theorems in any context (e.g. Rings, K-Algebra, or Topological spaces).

- Product in $\mathbb{R}$:

- Vector Space Bases:

- Haskell Type void (or some other cool example):

Lastly, we will provide a characterization of universality:

Proposition 2.1.1. Let $S : D \to C$ be a functor and $u : c \to Sr \in Ar(C)$. Then u is an universal arrow if and, only if, the function $\varphi : \hom_D(r, \cdot) \to \hom(c, S\cdot)$ such that $\varphi(d)(f) = Sf \circ u$ for all $f \in \hom_D(r, d)$ is a natural bijection. Conversely, every natural bijection is uniquely determined by an universal arrow $u : c \to Sr$.

Proof. Lets start by supposing that u is universal. Then for φ to be universal the diagram

$$\begin{tikzcd} \hom_D(r,d) \arrow[r, "\varphi(d)"] \arrow[d, "\hom_D(r,g)"'] & \hom_C(r,Sd) \arrow[d, "\hom_C(r,Sg)"] \\ \hom_D(r,d') \arrow[r, "\varphi(d')"] & \hom_C(r,Sd') \end{tikzcd}$$

should commute for all $g \in Ar(D)$. As this is a diagram in the category of sets, we can check the commutativity by checking it element wise. For any $f \in \hom_D(r, d)$:

$$\begin{tikzcd} f \arrow[r, "\varphi(d)"] \arrow[d, "\hom_D(r,g)"'] & Sf \circ u \arrow[d, "\hom_C(r,Sg)"] \\ g \circ f \arrow[r, "\varphi(d')"] & S(g \circ f) \circ u = Sg \circ Sf \circ u \end{tikzcd}$$

So the diagram commutes, and φ is natural. The bijectivity follows from the definition of u being universal.

Lets consider now that φ is a natural bijection. We will define $u := \varphi(r)(1_r)$ and check that (r, u) is an universal arrow. As φ is natural we have that:

$$\begin{tikzcd} \hom_D(r,r) \arrow[r, "\varphi(r)"] \arrow[d, "\hom_D(r,g)"'] & \hom_C(r,Sr) \arrow[d, "\hom_C(r,Sg)"] \\ \hom_D(r,d) \arrow[r, "\varphi(d)"] & \hom_C(r,Sd) \end{tikzcd}$$

Writing the diagram for the element $1_r \in \hom_D(r,r)$ and any $d \in Ob(D), g : r \to d \in \hom_D(r,d)$:

$$\begin{tikzcd}
1_r \arrow[rr, "\varphi(r)"] \arrow[d, "\hom_D(r,g)"'] & & u \arrow[d, "\hom_C(r,Sg)"] \\
g \arrow[r, "\varphi(d)"] & \varphi(d)(g) = Sg \circ u &
\end{tikzcd}$$

and since φ is a bijection, for every $f \in \hom_C(r, Sd)$ there is an unique function $f' = \varphi(d)^{-1}(f)$ such that $Sf' \circ u = f$, thus being u universal.

□

From this theorem there is a definition that arises:

Definition 2.1.2. Let D be a category with small home-sets and let $F : D \to C$ be a functor. A representation of a functor $K : D \to Set$ is a pair (r, φ) with $r \in Ob(D)$ and φ a natural isomorphism such that

$$D(r, \cdot) \equiv_\varphi F.$$

A functor is representable if it has a representation.

Note that therefore a universal arrow induces a natural isomorphism $D(r,d) \equiv C(c, Sd)$ and this induces a representation of the functor $C(c, S\cdot) : D \to Set$.

2.1.1 Yoneda's lemma

In this subsection deals with the Yoneda's Lemma. Mac Lane[3] assures the lemma first appeared in his private communication with Yoneda in 1954. With time, this result has became one of the most relevant one in Category Spaces. We will start by providing some intuition to it, followed by it proof and some use cases.

This results is due to japanese professor Nobuo Yoneda. We know about Yoneda's life thanks to the elegy that was written by Yoshiki Kinoshita[7]. Yoneda was born in Japan in 1930, and received his doctorate in mathematics from Tokyo University in 1952. He was a reviewer for international mathematical journals. In addition to his contributions to the field of mathematics, he also devoted his research to computer science.

The idea behind the yoneda lemma can be arid at first, if one does not have a prior understanding of what the purpose and usefulness of this lemma is. In order to ilustrate this idea we will introduce a (simplified) definition of Moduli Spaces, so that we have a geometric understanding of Yoneda's Lemma.

The idea behind (some) Moduli spaces is to classify algebraic curves up to isomorphisms. In addition, Moduli spaces allow us to control "complicated" mathematical objects (such as a quotient space of unknown objects) by simpler objects or objects with better properties (such as a concrete variety). A canonical example of this type of classification is:

$$\begin{aligned}\{\text{Vector spaces of finite dimension}\}/\text{isomorphism} &\cong \mathbb{N} \\ [V] &\to dimV.\end{aligned}$$

Where a complex object can be classified by another object of which we know more properties. We can start by defining:

$$\mathcal{M} = \{\text{smooth complex non singular curves}\}/\text{isomorphism}$$

Further on when we talk about curves we will refer to smooth complex non singular curves. Note that if two curves are isomorphic then they have the same genus. Therefore the function

$$\begin{aligned}\gamma : \mathcal{M} &\mapsto \mathbb{N} \\ [V] &\mapsto \text{genus of V}\end{aligned}$$

is well defined and we can define $\mathcal{M}_g = \gamma^{-1}(g)$. An interesting classification of $\mathcal{M}_g$ is given when we consider that for every g there exists a closed, connected, non-singular variety U_g and a family $\{C_t : t \in U_g$ such that a curve of genus g will be a fibration of C_t. Moreover there is a variety M_g and a superjjective morphism $\varphi : U_g \to M_g$ such that $\varphi(t_1) = \varphi(t_2)$ if $C_{t_1} \equiv C_{t_2}$. Therefore we are classifying the equivalence classes of $\mathcal{M}_g$ by points of the variety M_g (thus generating a Moduli problem).

Similarly to this two example, with the Yoneda lemma we will have a functor $F : D \to Set$ and one representation of this functor. We will classify the natural transformation of these functors by the set in the image of F! Interestingly enough, there will be applications where the complex object is not the space of naturals transformations, but the images of F (see 2.1.2 for an example). Lets proceed to enunciate and proof the result.

Theorem 2.1.1. *[3, Section 3.2] Let D be a category with small home-sets, $F : D \to Set$ be a functor, and $r \in Ob(D)$. Then there is a bijection*

$$\begin{aligned}&\tau : Nat(\hom_D(r, \cdot), F\cdot) \equiv Fr \\ &\tau(\alpha : \hom_D(r, \cdot) \to F\cdot) = \alpha(r)(1_r)\end{aligned}$$

Where τ is natural in K(as an object of Set^D) and in r.

Proof. As α is a natural transformation we have that, for every $g : r \to d \in Ar(D)$:

$$\begin{tikzcd} \hom_D(r,r) \arrow[r, "\alpha(r)"] \arrow[d, "\hom_D(r,g)"'] & Kr \arrow[d, "Kg"] \\ \hom_D(r,d) \arrow[r, "\alpha(d)"] & Kd \end{tikzcd}$$

Writing $\alpha(r)(1_r) = u$ we have that:

$$\begin{tikzcd} 1_r \arrow[r, "\alpha(r)"] \arrow[d, "\hom_D(r,g)"'] & u \arrow[d, "Kg"] \\ g \arrow[r, "\alpha(d)"] & \alpha(d)(g) = Kg(u) \end{tikzcd}$$

Therefore every natural transformation is uniquely identified by the value of u, therefore τ is injective. Moreover, for every u in Kr, we can define a natural transformation following the previous diagram, therefore, τ is bijective.

To see that τ we is natural we have to consider for which functor it is natural. Consider the functor *Evaluation* $E : Set^D \times D$ that maps each $(F, c) \to Fc$, and the functor $N : Set^D \times D \to Nat(\hom_D(r, \cdot), K)$ the set of natural transformations. Finally, $\tau : N \to E$ is a natural transformation. $\square$

The first functor that we will want to apply this result is to the hom. functor. But this functor is a bifunctor, so to get the full result of this lemma applied to the full bifunctor we may restate this lemma to contravariant functors.

Corollary 2.1.1.1. *Let D be a category with small home-sets, $F : D \to Set$ be a contravariant functor, and $r \in Ob(D)$. Then there is a bijection*

$$\tau : Nat(\hom_D(\cdot, r), F\cdot) \equiv Fr$$
$$\tau(\alpha : \hom_D(\cdot, r) \to F\cdot) = \alpha(r)(1_r)$$

Where τ is natural in K(as an object of Set^D) and in r.

Proof. We seek to use the Yoneda lemma in the functor $F' : D^{op} \to Set$ induced by F. Then we have that:

$$\tau : Nat(\hom_{D^{op}}(r, \cdot), F'\cdot) \equiv F'r$$
$$\tau(\alpha : \hom_{D^{op}}(r, \cdot) \to F'\cdot) = \alpha(r)(1_r)$$

Taking into account that $F_{|Ob(D)} = F'_{|Ob(D^{op})}$, and that $\hom_{D^{op}}(r, \cdot) = \hom_D(\cdot, r)$ we have the result. $\square$

As we have seen, the Yoneda lemma is a direct generalization of the moduli problem. In the same vein, yoneda's lemma is the generalisation of other problems/theorems in mathematics, most notably Cayley's lemma. It states:

Proposition 2.1.2. Any group is isomorphic to a subgroup of a symmetric group.

To understand this, take a groups G seen as a single-object category, and name that object e. Then, the functor $\hom_G(e, \cdot) : G \to Set$ can be seen as a group action 1.3.2. Then the Yoneda lemma states that:

$$Nat(\hom_G(e, \cdot), Hom_G(e, \cdot)) \equiv_\varphi Hom_G(e, e).$$

Translating this result to group theory:

- Remember that $Hom_G(e,e)$ is the group G.
- Every natural transformation is a equivariant map between G-sets.
- This equivariant maps, forms an endormophism group under composition, being a subgroup of the group of permutations.
- This natural isomorphism φ define a group isomorphism.

So we have the isomorphism of groups that is stated in Cayley's Theorem.

We continue our exploration of Yoneda lemma by defining the *Yoneda Embedding*. For that we define the contravariant functor $h_a = \mathrm{hom}_C(\cdot, a)$. Then the contravariant Yoneda lemma tell us that:

$$Nat(h_a, h_b) \equiv_{\tau_a} \mathrm{hom}(a,b).$$

We then can define a fully faithfull embedding $v : C \to Set^{C^{op}}$ such that

$$\begin{aligned} va &= \mathrm{hom}_C(A, \cdot) && \forall a \in Ob(C), \\ vf &= \tau_a^{-1}(f) && \forall f : b \to a \in Ar(C). \end{aligned}$$

This functor allows us to view the category C as a subcategory of the category of contravariant functors from C to Set, which will be useful for determining "heritable" properties in C.

2.1.2 Properties expressed in terms of universality

After the examples given, we will define a few constructions that are present in various parts. We will outline the notions of limit, pullback and product, and the dual notions of colimit, pushout and coproduct.

The notions of product and pullback can be seen as particular cases of the notion of limit. We will therefore begin by defining this concept as an introductory step. In turn, to define limit we will introduce the concept of co-cone and the diagonal functor.

Definition 2.1.3. Let C, J be categories. We can define the functor $\Delta_J : C \to C^J$ that maps c to the functor from J to C that is constantly c, and maps every arrow to the identity 1_c.

Whenever possible we will write only Δ, and let the information of the category be deduced from context. J is usually small and often finite. We can now consider a natural transformation $\tau : F \to \Delta c$. This can be represented as in the following diagram:

$$\begin{tikzcd} Fx_j \arrow[rr, "Fg"] \arrow[dr, "\tau x_j"'] & & Fx_k \arrow[dl, "\tau x_k"] \\ & c & \end{tikzcd}$$

commutes for every $g : x_j \to x_k \in Ar(j)$, for that reason, such natural transformation is usually called a co-cone. The dual notion is called cone and is represented as:

$$\begin{tikzcd} Fx_j \arrow[rr, "Fg"] & & Fx_k \\ & c \arrow[ul, "\tau x_j"] \arrow[ur, "\tau x_k"'] & \end{tikzcd}$$

We can now define the concept of limit and colimit. We introduce first the concept of colimit. This definition is a basic definition of a universal arrow, only in a category of functors. Following this definition, we will define the limit as a dual concept.

Definition 2.1.4. A colimit is an object $r \in Ob(C)$ together with an universal arrow $u : F \to \Delta r \in Ar(C^J)$. The colimit is denoted by

$$\lim_{\leftarrow} F = r = \operatorname{colim} F.$$

The notation $\lim_{\leftarrow}$ is intuitive as we can see that in the colimit we have arrows to F. To represent this as a diagram, we have a co-cone $u \to \operatorname{colim} F$ such that for every other co-cone $\tau \to s$, it exist an unique f such that the following commutes for every $x_j, x_k \in Ob(C)$:

$$\begin{tikzcd} & l & \\ & \operatorname{colim} F \arrow[u, dashed, "f"] & \\ x_j \arrow[uur, bend left, "\tau x_j"] \arrow[ur, "u x_j"] \arrow[rr, "Fg"'] & & x_k \arrow[uul, bend right, "\tau x_k"'] \arrow[ul, "u x_k"'] \end{tikzcd}$$

Now, thanks to the duality of categories, we can define what a limit is in a very synthetic way:

Definition 2.1.5. A limit is the dual concept of a colimit. It is denote as

$$\lim_{\rightarrow} F = r = \lim F.$$

A limit is represented by the following diagram:

$$\begin{tikzcd} & l \arrow[d, dashed, "f"] \arrow[ddl, bend right, "\tau x_j"'] \arrow[ddr, bend left, "\tau x_j"] & \\ & \lim F \arrow[dl, "u x_j"'] \arrow[dr, "u x_k"] & \\ x_j \arrow[rr, "Fg"'] & & x_k \end{tikzcd}$$

Analogously as in the colimit notation, in the limit we have arrows from F, and thus the notation $\lim_{\rightarrow}$. Let focus for a while now in the notion of limit, in particular of two of its specials cases: the product and the pullback. From these cases we are going to provide most examples.

We have already talked about the product of categories, and in example 2.1.1 we denote that in these type of construction there is some sort of universality involved. The product is a limit when J is the 2-element discrete category, that is, when every functor from $J \to C$ is merely choosing to object of C.

Definition 2.1.6. Let C be a category, $J = \{0, 1\}$ be the discrete category with two elements. The product $c_1 \times c_2$ of two elements $c_0, c_1 \in Ob(C)$ is the limit of the functor $F : J \to C$ such that $F0 = c_0, F1 = c_1$.

This construction means that providing an arrow to c_0, c_1 determines an unique arrow to $c_0 \times c_1$. In this case, the arrows $u0, u1$ are usually called *projections* and denoted by π_0, π_1. The notation is due to the product being a generalization of the cartesian product in the category *Set*. It is also sometimes note with $c_0 \Pi c_1$ with Π being stardard for the sequence product. Some examples of product object in categories are:

Example 2.1.2.

- Product of Banach spaces:
- Product of something in haskell.

Analogously, we can define the coproduct, on which instead of defining an arrow to c_0, c_1, we define an object *from* c_0, c_1.

Definition 2.1.7. The coproduct is the dual definition of the product. It is denoted by $c_0 \sqcup c_1$.

In this notation $\sqcup$ denotes an inverted Π, with the meaning of being the dual notion of the product.

Example 2.1.3.

- Free product of groups
- Union of enumerate types.

From my personal experience I have to say that I have seen more difficulties learning this notion rather than learning the utterly similar notion of product, probably because we are more used to think in terms of arrays rather than in terms of universality for the product. I think that thinking in terms of universality should be the way in to these concept.

After learning about the product and the coproduct, we will focus now in the notion of pullback and its dual, the pushout. We will first define the category

$$P = \quad x \xrightarrow{f} z \xleftarrow{g} y$$

Then we can define the pullback:

Definition 2.1.8. Let C be a category and $F : P \to C$ be a functor. Then the pullback of Fx and F_y denoted as $Fx \times_{Fz} Fy$ is the limit of the functor F.

We can represent this structure in the following diagram. For any other object q and arrows $f' : q \to Fx, g' : q \to Fy$ we have:

$$\begin{tikzcd}
q \arrow[drr, bend left, "q_2"] \arrow[dr, dashed, "u"] \arrow[ddr, bend right, "q_1"'] & & \\
& Fx \times_{Fz} Fy \arrow[r, "p_2"] \arrow[d, "p_1"'] & Fy \arrow[d, "Fg"] \\
& Fx \arrow[r, "Ff"'] & Fz
\end{tikzcd}$$

Analogously, we can define:

$$CoP = \begin{tikzcd} x & z \arrow[l, "f"] \arrow[r, "g"'] & y \end{tikzcd}$$

and define:

Definition 2.1.9. Let C be a category and $F : CoP \to C$ be a functor. Then the pushout of Fx and F_y denoted as $Fx \sqcup_{Fz} Fy$ is the colimit of the functor F.

Example 2.1.4.
- Fiber bundles (pullback)
- Suppose that X, Y, and Z as above are sets, and that f : Z → X and g : Z → Y are set functions. The pushout of f and g is the disjoint union of X and Y, where elements sharing a common preimage (in Z)
- Seifert-Van-Kampen (Si tienes valor).

We can regard that there are an evident similarity on the notation of the pullback/pushout and the one of product/coproduct. To understand these we have to consider the similarities of both construction. We are going to focus on the similarities of product and pullback.

In both case, the universal property consists of having an arrow to a generated object only if we have an arrow to each of its generators. In this line of reasoning, we can consider that the product is a pullback where we forget about the object z and its arrows. One easy way to generate that case is to consider the construction when Fz is a terminal object. In that case we have that the existence of Ff and Fg is a tautology, and we can only consider

$$\begin{tikzcd}
& q \arrow[dl, bend right, "q_1"'] \arrow[d, dashed, "u"] \arrow[dr, bend left, "q_2"] & \\
Fx & Fx \times_{Fz} Fy \arrow[l, "\pi_0"'] \arrow[r, "\pi_1"] & Fy
\end{tikzcd}$$

having a product structure. One can proceed to consider an analogous consideration with pushout and coproduct.

2.2 Adjoints

The adjoints is a fundamental property on Category theory. It relates two functors $F : C \to D, G : D \to C$. The importance of adjoints however, comes from it ubiquity among mathematics, often relate with the ubiquity of universality. This notion was first presented by Daniel M. Kan in [2]. For this section we follow [3].

We will start by providing some intuitive notions, before a formal definition. Take the Forgetful Functor $U : Grp \to Set$, and the Free Group Functor $\mathcal{F} : Set \to Grp$. We can see that it is not difficult to consider that we can compose F and G to make endofunctors. These endofunctors are far from being identities, nonetheless we have still a great relation between them: for every set X and group G, there is a function $X \to UG$ for each morphism $FX \to U$. Conversely, any function from X to UG induces a morphism.

Any avid reader will detect by this point the taste of universality. But there is a bit more, we have a bijective relation on every image home-set of both F and G. This is the underlying property of adjointness - to relate home-sets. Lets formalize this idea:

Definition 2.2.1. Let C, D be categories. And adjunction is a triple $(F, G, \varphi) : D \to C$ where F, G are functors:

$$\begin{tikzcd} D \arrow[r, shift left, "G"] & C \arrow[l, shift left, "F"] \end{tikzcd}$$

while φ is a transformation that maps each $(c, d) \in Ob(C \times D)$ with a bijection:

$$\varphi_{c,d} : \hom_C(Fd, c) \equiv \hom_D(d, Gc).$$

which is natural in c and d.

We have a bit to unpack in this definition. We will start to understand in what sense φ is natural. Remembering about N functor in Yoneda's Lemma, we can see that it is natural bijection from $\varphi : Hom_C(F\cdot, \cdot) \to \hom_D(\cdot, G\cdot)$ in each variable. That is, for every $f : d \to d' \in C, k : c \to c' \in D$ the diagrams

$$\begin{tikzcd} \hom_C(Fd,c) \arrow[r, "\varphi_{c,d}"] \arrow[d, "\hom_C(Fd,\cdot)(g)"] & \hom_D(d,Gc) \arrow[d, "\hom_D(d,G\cdot)(g)"] \\ \hom_C(Fd,c') \arrow[r, "\varphi_{c',d}"] & \hom_D(d,Gc') \end{tikzcd} \qquad \begin{tikzcd} \hom_C(Fd,c) \arrow[r, "\varphi_{c,d}"] \arrow[d, "\hom_C(F\cdot,c)(f)"] & \hom_D(d,Gc) \arrow[d, "\hom_D(\cdot,Gc)(f)"] \\ \hom_C(Fd,c') \arrow[r, "\varphi_{c,d'}"] & \hom_D(d,Gc') \end{tikzcd}$$

comutes.

We can follow by considering the important notation decision: how to call the adjoints. This is a quite ambiguous notation, but we will stick with the notation of F being the *left adjoint* and G being the *right adjoint*. When a functor *is* a left (resp. right) adjoint it *has* a right (resp. left) adjoint. This notation comes from where is the functor placed in the home-set, when making the bijection.

We have already suggested the relationship between universality and adjointness. We can summary that relation in the following property:

Proposition 2.2.1. An adjunction $(F, G, \varphi) : D \to C$ determines:

- a natural transformation $\eta : I_D \to GF$ such that $\eta(x) : x \to G$ is universal for every $x \in Ob(D)$. Conversely, we can define:

$$\varphi(f : Fd \to c) = Gf \circ \eta(d) : d \to Gc.$$

- a natural transformation $\epsilon : FG \to I_C$ such that $\epsilon(x) : Fx \to a$ is universal for every $x \in Ob(C)$. Conversely, we can define:

$$\varphi(g : d \to Gc) = \eta(c) \circ Fg : Fd \to c.$$

Proof. TODO □

From this proposition we can see that we may be able to define an adjoint based only on the universal arrows provided. We can summary a few equivalent definition of adjoint:

Proposition 2.2.2. Each adjoint is completely determined by the items in any one of the following list:

1. TODO

To further ilustrate this concept we present some examples:

Example 2.2.1. • TODO

2.2.1 Monad

2.2.2 Algebra

2.2.3 Closed Cartesian Categories

No se muy bien donde meterlo aún.

Bibliography

[1] Stefan Banach. "Théorie des opérations linéaires". In: (1932).

[2] Daniel M Kan. "Adjoint functors". In: *Transactions of the American Mathematical Society* 87.2 (1958), pp. 294–329.

[3] Saunders Mac Lane. *Categories for the working mathematician*. Vol. 5. Springer Science & Business Media, 2013.

[4] Bartosz Milewski. *Category theory for programmers*. Blurb, 2018.

[5] Emily Riehl. *Category theory in context*. Courier Dover Publications, 2017.

[6] Sigur. *Using Tikz-cd to Illustrate Composition of Natural Transformations*. Computer Science Stack Overflow. URL: https://tex.stackexchange.com/questions/505322/using-tikz-cd-to-path-to-illustrate-composition-of-natural-transformations%20(version:%202021-02-05).

[7] Kinoshita Yoshiki. "Prof. Nobuo Yoneda passed away". In: (1996). URL: https://www.mta.ca/~cat-dist/catlist/1999/yoneda.